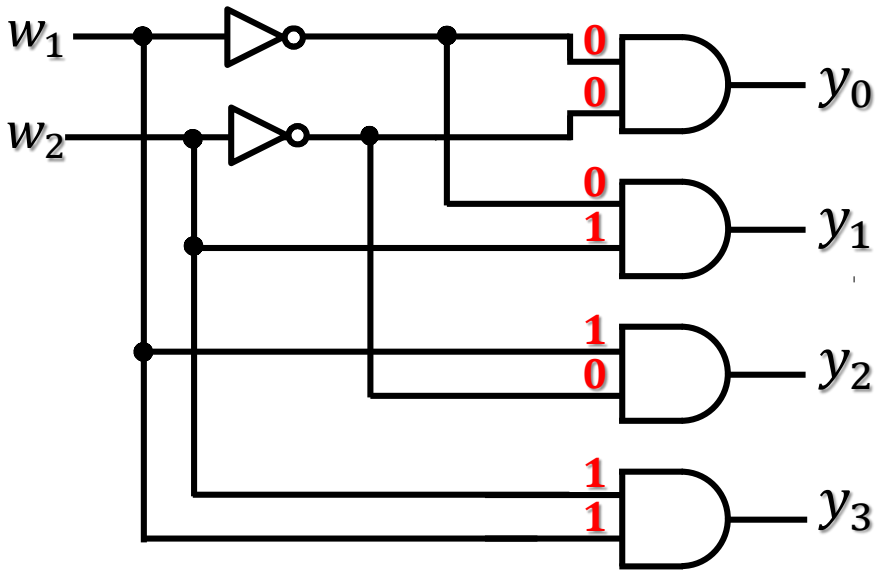
Decoders

- Like MUX's, another standard building block is the decoder.
- Decoder circuits
  - Used to decode encoded information.
- A binary decoder
  - Logic circuit with n inputs and 2^n outputs.
- Only one output is asserted (1) at a time,
  - Each output corresponds to 1 valuation of the inputs.
- The outputs of a decoder can be considered as "one-hot" encoded.

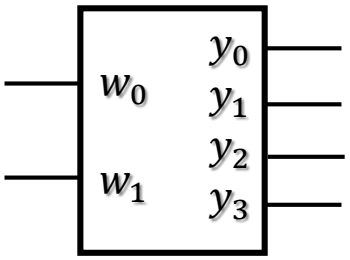
Decoders

- Consider
  - 2 inputs w_0, w_1 and 4 outputs y_0, y_1, y_2, y_3
  - Only 1 output asserted at a time
  - *"Binary Decoder"*

w_1	w_2	y_0	y_1	y_2	y_3
0	0	**1**	0	0	0
0	1	0	**1**	0	0
1	0	0	0	**1**	0
1	1	0	0	0	**1**



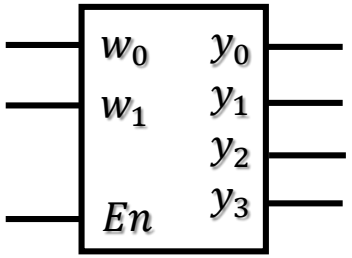
Logic circuit



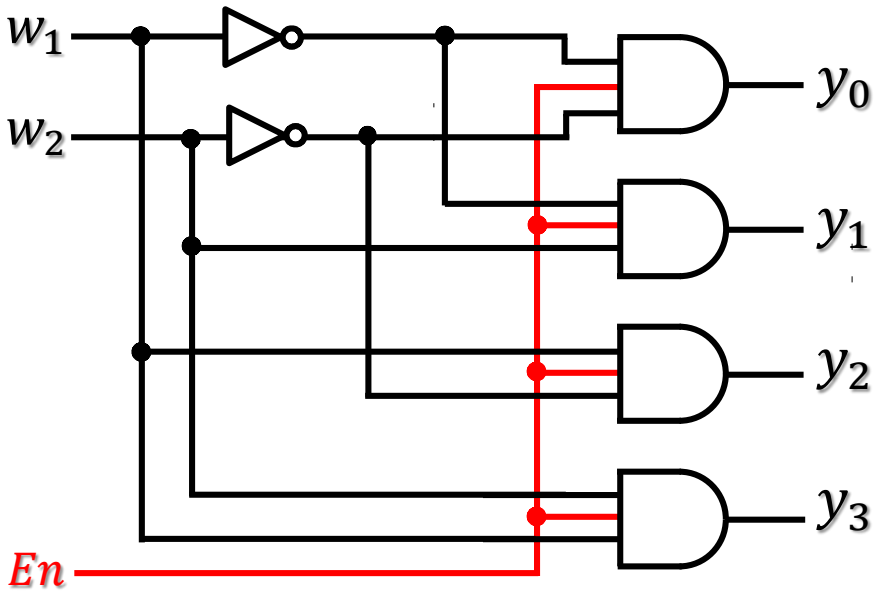
Graphical symbol

Decoders

- *"Binary Decoder"*
  - Each output - driven by an **AND** gate
    - Decodes valuation of w_1, w_2
    - "*Enable*" input is useful



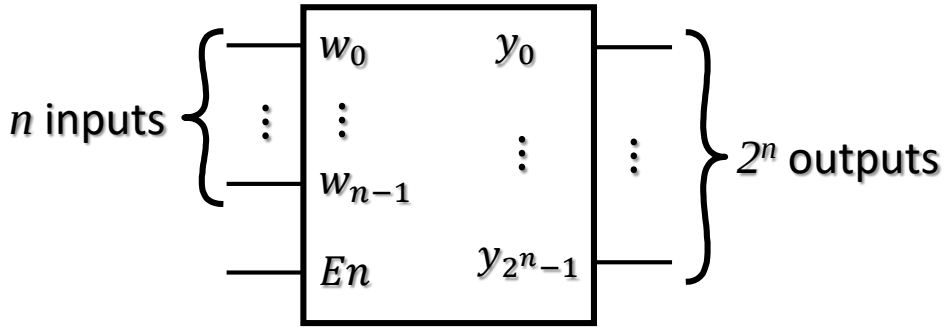
Graphical symbol



En	w_1	w_2	y_0	y_1	y_2	y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Decoders

- Generic n to 2^n *"Binary Decoder"*

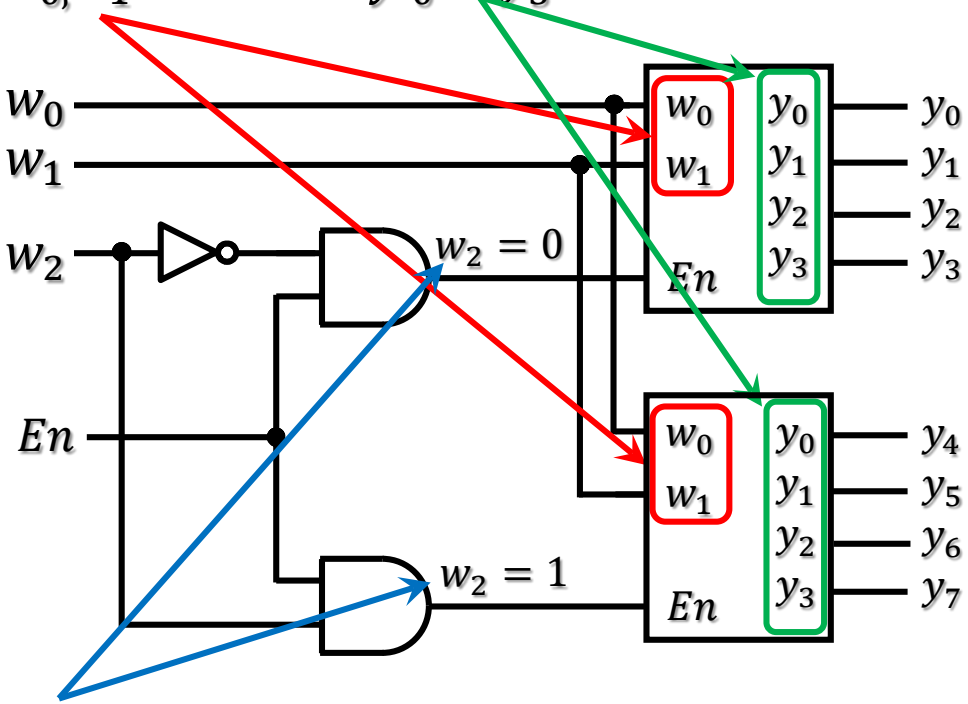


A n to 2^n decoder

- A k bit binary code – exactly 1 bit set at a time
  - Referred to a *"one-hot encoded"*
- Outputs of a binary decoder are *"one-hot encoded"*
- Larger decoders can be constructed using
  - Using sum-of products (as seen previously) or
  - Using groups of smaller decoders

Decoders: 3-to-8 decoder

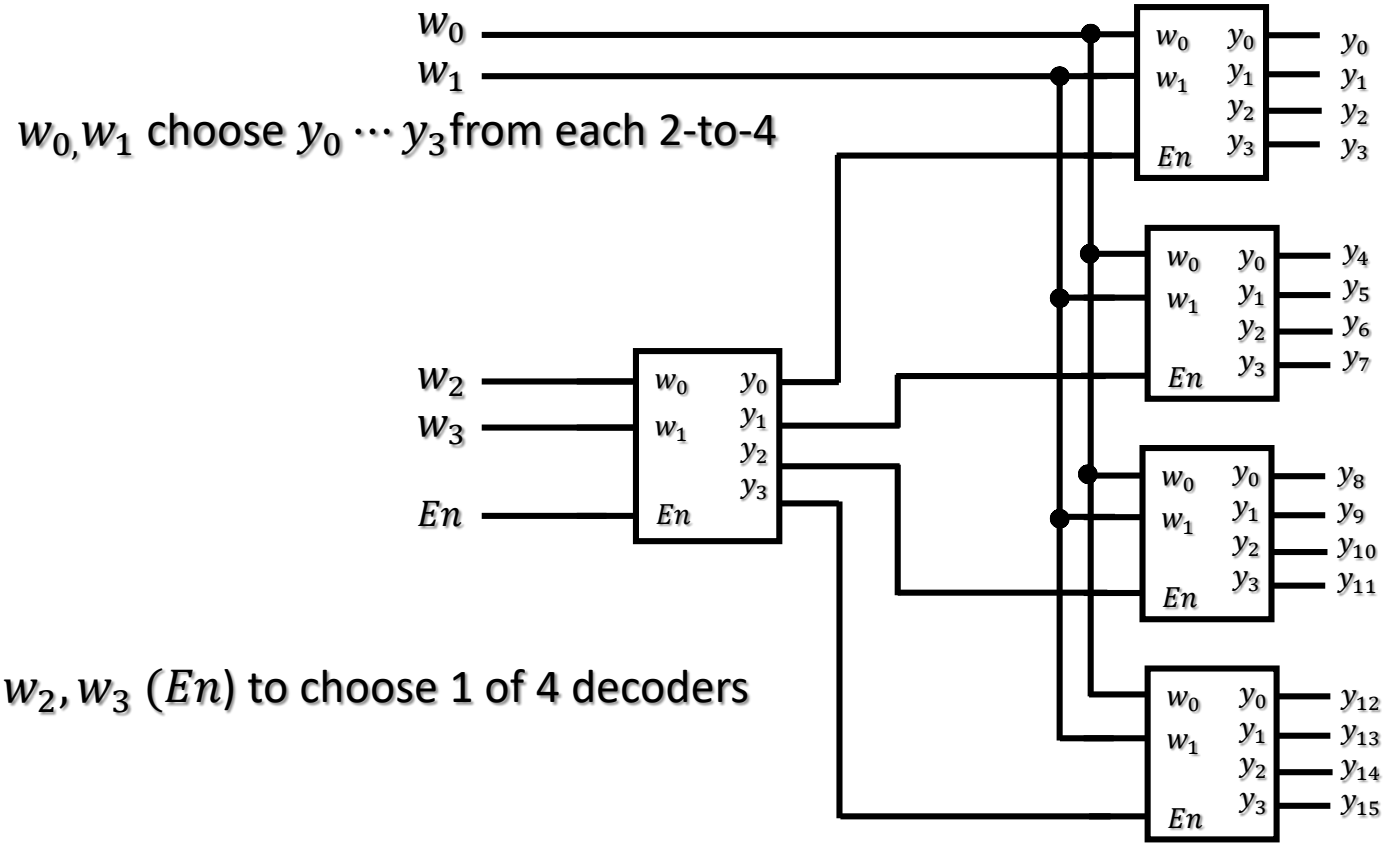
- Example
  - A 3-to-8 (inputs w_0, w_1, w_2) decoder using two 2-to-4 decoders
  - Use w_0, w_1 to choose $y_0 \cdots y_3$ from each 2-to-4



  - Use w_2 to choose between the two decoders (with En)

Decoders

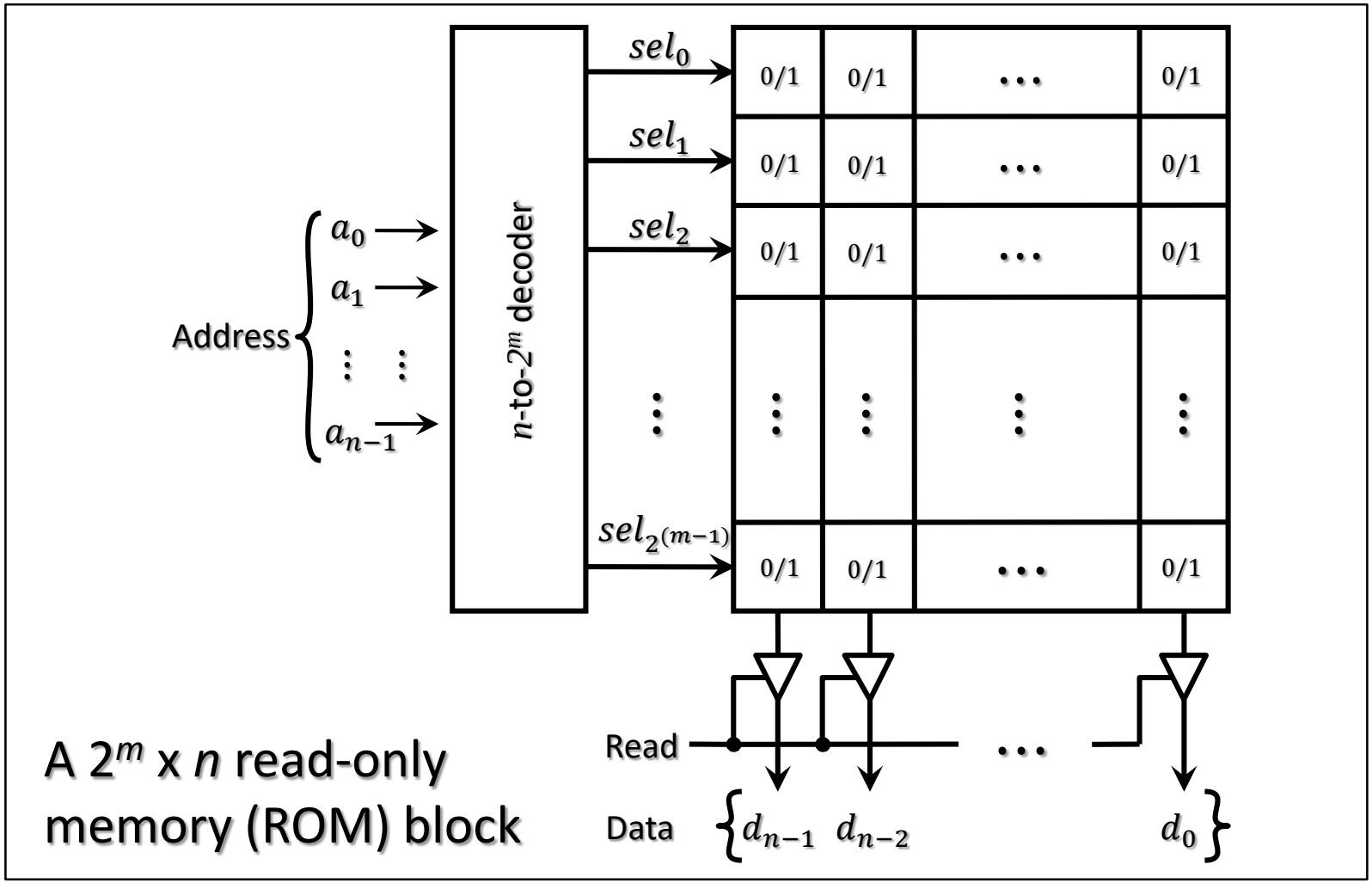
- Example - 4-to-16 decoder - (5) 2-to-4 decoders



Decoders – in memory

- One of the most important applications of decoders is in addressing memory blocks.
  - Such memory blocks are included in many digital systems
  - One type of memory block - a read-only memory (ROM).
    - Collection of storage cells
    - Each cell permanently stores a single logic value, 0 or 1.
    - Stored information can be read out of the storage cells, but it cannot be changed.
  - Programmable ROM (PROM)
    - Allows information to be both – read and write
      - Read out of the storage cells and stored, or
      - Written (generally one time only), into them.
- Many different types of memory blocks exist.

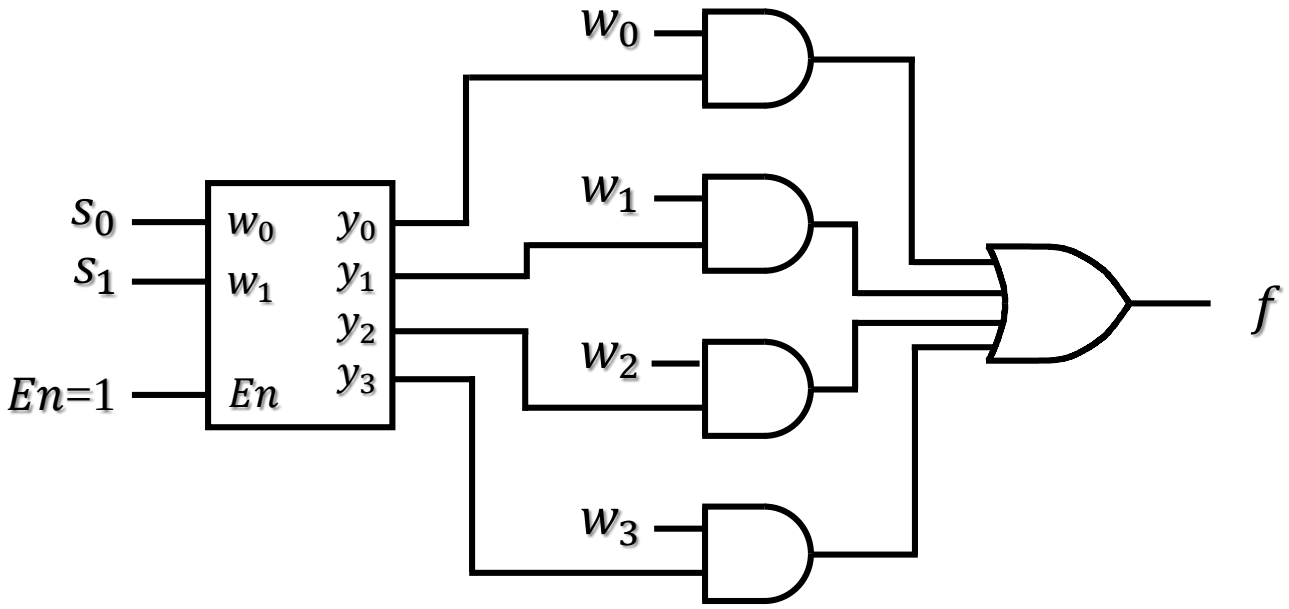
Decoders – ROM decoder



A 2^m x n read-only memory (ROM) block

4-to-1 multiplexer – decoder

- Decoder evaluates the values on its inputs
  - Can be used to build a multiplexer.

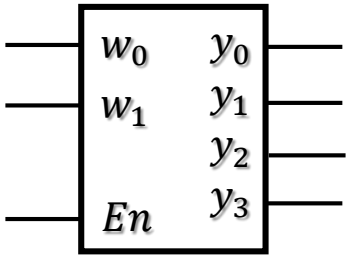


De-Multiplexer

- Multiplexer
  - n data inputs and $\log_2 n$ select inputs
  - Multiplex n data inputs onto one single output
    - Under control of select inputs
- Demultiplexer
  - Opposite function
  - Place value of a single data input on multiple outputs
  - Can be implemented using a decoder circuit

Demultiplexer

- For a 1-to-4 demultiplexer
  - A 2-to-4 decoder can be used
  - En used as data in
  - y_0 to y_3 as data out
  - Valuation of w_0, w_1 determines
    - which output set to En
  - When $En = 0$, all outputs $= 0$
    - w_0, w_1 - which (valid) output $= 0$
  - When $En = 1$
    - w_0, w_1 - which output $= 1$
  - In general an n to 2^n decoder
    - Can be used as a 1 to n demultiplexer
    - In practice decoder circuits used more often



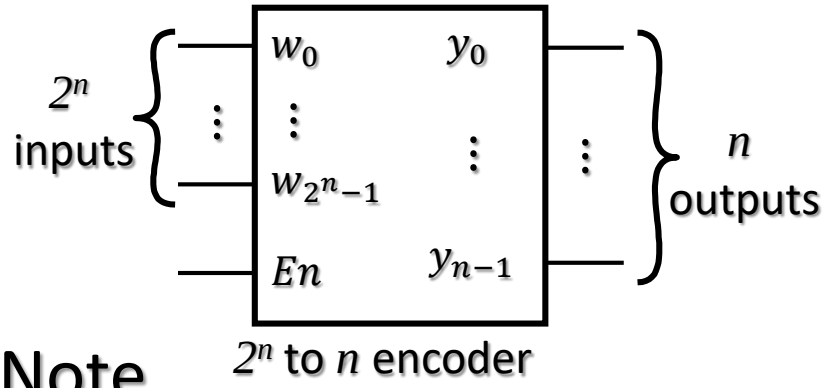
Graphical symbol

En	w_1	w_2	y_0	y_1	y_2	y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Encoders

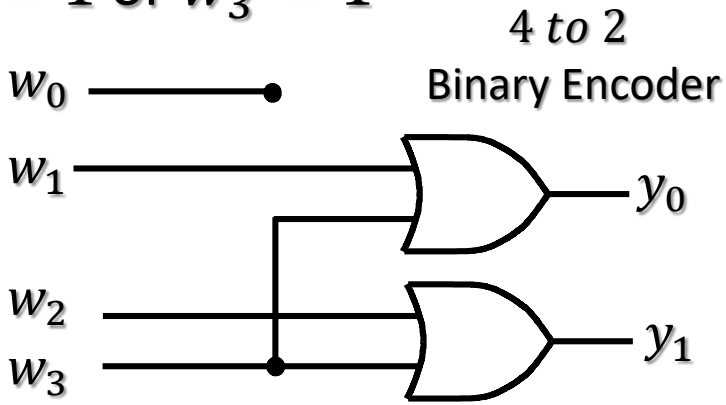
- Encoders
  - Encoder performs the opposite function of a decoder.
  - Encodes given information into a *more compact form*.
  - Many ways information is encoded, BCD, Grey code, etc.
- Binary Encoders
  - A binary encoder encodes information from $2n$ inputs into an n -bit code.
  - Only one of the input signals should have a value of 1
  - Then the outputs present the binary number which corresponds to the input which is equal to 1.

Encoders



w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- Note
  - $y_0 = 1$ when either $w_1 = 1$ or $w_3 = 1$
  - $y_1 = 1$ when either $w_2 = 1$ or $w_3 = 1$
- Inputs –
  - "one-hot" encoded
  - Reduce # bits needed
  - Fewer bit/wires



Priority Encoders

- Each input has an associated priority level
  - Output identifies the input with highest priority.
  - Input with a high priority is asserted
    - Inputs with lower priority are ignored
  - Assumption
    - Input w_0 has lowest priority
    - Input w_3 has highest priority
  - Valid output z=0 if all inputs =0
    - Output y_1 and y_0 not meaningful
  - Output z=1 if at least one =1

Truth table for a 4-to-2 priority encoder

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	**1**	0	0	1
0	0	**1**	x	0	1	1
0	**1**	x	x	1	0	1
1	x	x	x	1	1	1

Priority Encoders

- Truth table – synthesized as before but …..
- Define a set of intermediate signals - $i_0 \cdots i_3$ based on "priority observations"

$$i_0 = \overline{w_3}\,\overline{w_2}\,\overline{w_1}w_0$$
$$i_1 = \overline{w_3}\,\overline{w_2}w_1$$
$$i_2 = \overline{w_3}w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$
$$z = i_0 + i_1 + i_2 + i_3$$

Truth table for a 4-to-2 priority encoder

	w_3	w_2	w_1	w_0	y_1	y_0	z
	0	0	0	0	d	d	0
i_0 →	0	0	0	**1**	0	0	1
i_1 →	0	0	**1**	x	0	1	1
i_2 →	0	**1**	x	x	1	0	1
i_3 →	**1**	x	x	x	1	1	1